LA-UR- 03-1931

Approved for public release; distribution is unlimited.

Title:

An Electronic Notebook for Physical System Simulation

Author(s): Robert L. Kelsey

Submitted to: SPIE AeroSense
April 21-25, 2003
Orlando, FL

99

An electronic notebook for physical system simulation

Robert L. Kelsey

Los Alamos National Laboratory
X-8 MS-F645, Los Alamos, NM 87545

ABSTRACT

A scientist who sets up and runs experiments typically keeps notes of this process in a lab notebook. A scientist who runs computer simulations should be no different. Experiments and simulations both require a set-up process which should be documented along with the results of the experiment or simulation. The documentation is important for knowing and understanding what was attempted, what took place, and how to reproduce it in the future.

Modern simulations of physical systems have become more complex due in part to larger computational resources and increased understanding of physical systems. These simulations may be performed by combining the results from multiple computer codes. The machines that these simulations are executed on are often massively parallel/distributed systems. The output result of one of these simulations can be a terabyte of data and can require months of computing. All of these things contribute to the difficulty of keeping a useful record of the process of setting up and executing a simulation for a physical system.

An electronic notebook for physical system simulations has been designed to help document the set up and execution process. Much of the documenting is done automatically by the simulation rather than the scientist running the simulation. The simulation knows what codes, data, software libraries, and versions thereof it is drawing together. All of these pieces of information become documented in the electronic notebook. The electronic notebook is designed with and uses the eXtensible Markup Language (XML). XML facilitates the representation, storage, interchange, and further use of the documented information.

Keywords: physical system simulation, electronic notebook, XML-based representation

1. BACKGROUND

1.1. Simulations

The size and complexity of simulations has grown with the size and complexity of computing machines. It is no longer enough for physical systems to be simulated in two dimensions at coarse resolution. Now, physical systems must be simulated in three dimensions with increasingly finer resolution to gain insight into more and smaller details. A greater amount of input information is used to initialize these simulations and a higher fidelity result is expected. All of this has contributed to the increasing complexity of simulations.

The increase in size and complexity of simulations has in turn led to an increase in size and complexity of the machines executing the simulations. "The larger the computer, the greater the complexity that could be simulated accurately."[1] There has been an evolution of larger machines, faster processors, increased number of processors, increasingly complex interconnection networks, and more sophisticated parallel programming strategies to reap answers from simulation problems.

Typical physical system simulations of yesterday at Los Alamos consisted of approximately two input files. Execution of the simulation resulted in approximately 50 output files. The run time was a matter of hours with overnight turnaround. The physical system simulations of today consist of one or more input files often larger in size than yesterday's input file. It can take weeks to months of run time for a single simulation to complete execution. The result can be thousands of output files.

Further author information: E-mail: rob@lanl.gov

In this application there are a number of separate XML-based representations being used. However, the main representation is the representation of a notebook. In the representation the root element is the enotebook element. The enotebook element has a date attribute which identifies the creation date of the notebook and a name attribute which is a unique identifier of the notebook. An enotebook element can contain any number of entry elements. An entry element represents an entry in the notebook. Figure 1 shows an abstract view of the representation of a notebook. Elements are shown in a slightly larger type face and are connected by arrows. The attributes of an element are shown to the right of each element. The arrows show what child elements are contained in a parent element.

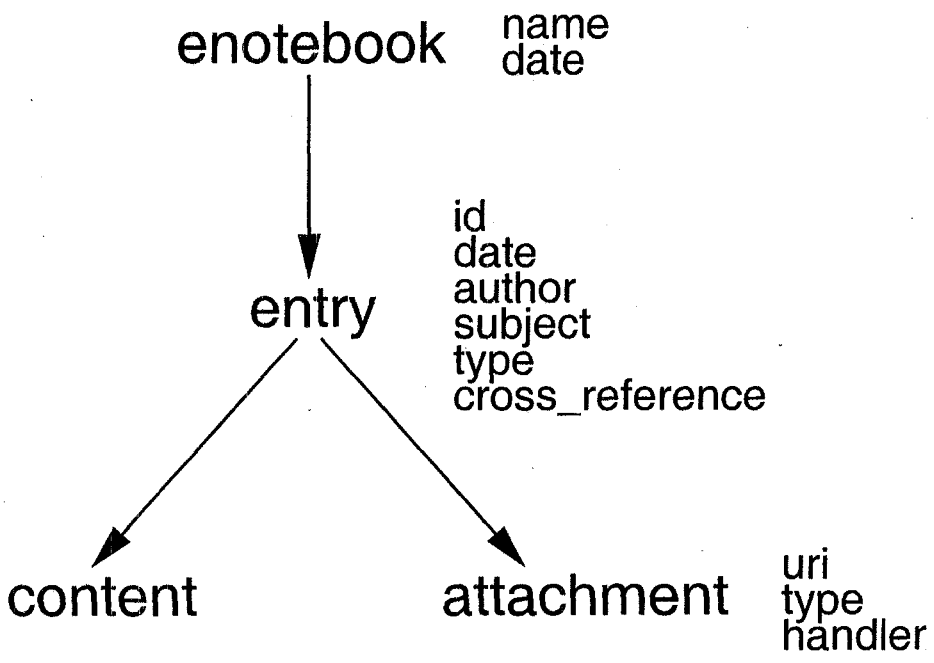

Figure 1. An abstract view of the representation of a notebook.

The entry element has six attributes: id, date, author, subject, type, and cross reference. The id is a unique identification of an entry. The date is the creation date of the entry. The author is who or what created the entry. The author could be a use or a computer program. The subject is a brief message about the topic of the entry. The type describes the category of the entry, such as a set up or execution entry. The cross reference is available to reference between entries. The entry element contains a content element and any number of attachment elements.

The content element has no attributes. The content element contains the message or note about the entry. The attachment element represents files or information to be attached to an entry. It can either contain or point to the file or information. The attachment element has three attributes: uri, type, and handler. The uri is a uniform resource identifier and serves as a pointer to an external file or resource. The type identifies what type or format the attachment is, for example, XML or text format. The handler identifies a program to read or view the attachment.

Another important XML-based representation is the representation for physical systems for simulation. This representation includes all the objects necessary to create a description of a physical system such as a shock wave physics problem or the motion of a projectile. At the highest level the representation includes objects for a computational mesh, simulation start up information, a set of materials, a set of geometric bodies, and simulation output information.[5,7] The representation also serves as a data-interchange format. This means that a single physical system description can be converted to several different native formats and simulated on several different application codes.

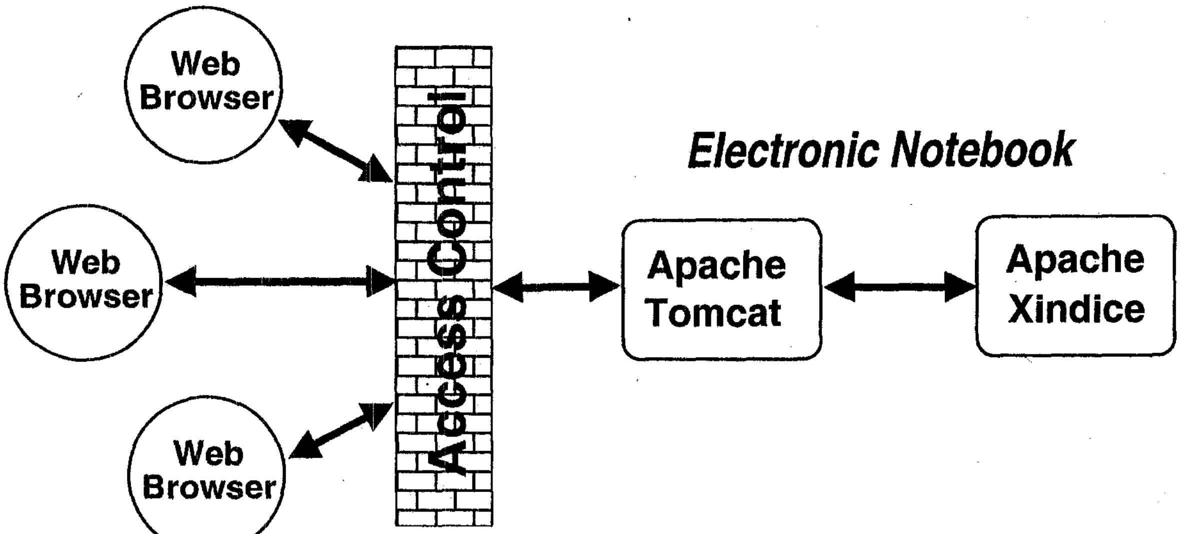

Figure 3. User interaction with the electronic notebook.

2.2.4. Program requests

Programs executing within simulations can make both read and write requests but as opposed to users, programs typically make mostly write requests to the notebook. Since there is no user involved (other than starting the simulation) the requests are not interactive. Thus, the path to the notebook is different and uses different technology. Like the user requests, program requests still go through the access control system.

Simulation programs use a notebook application programming interface (API) to send requests through the access control system (after authentication) to an XML Remote Procedure Call (XMLRPC) server. XMLRPC is a specification[14] for transporting XML encoded data via hypertext transfer protocol (HTTP). XMLRPC is particularly useful for tying together disparate systems and environments. Figure 4 diagrams how program requests are made to the electronic notebook. Multiple simulations are shown making requests. Each individual simulation uses the notebook API to connect to the notebook.

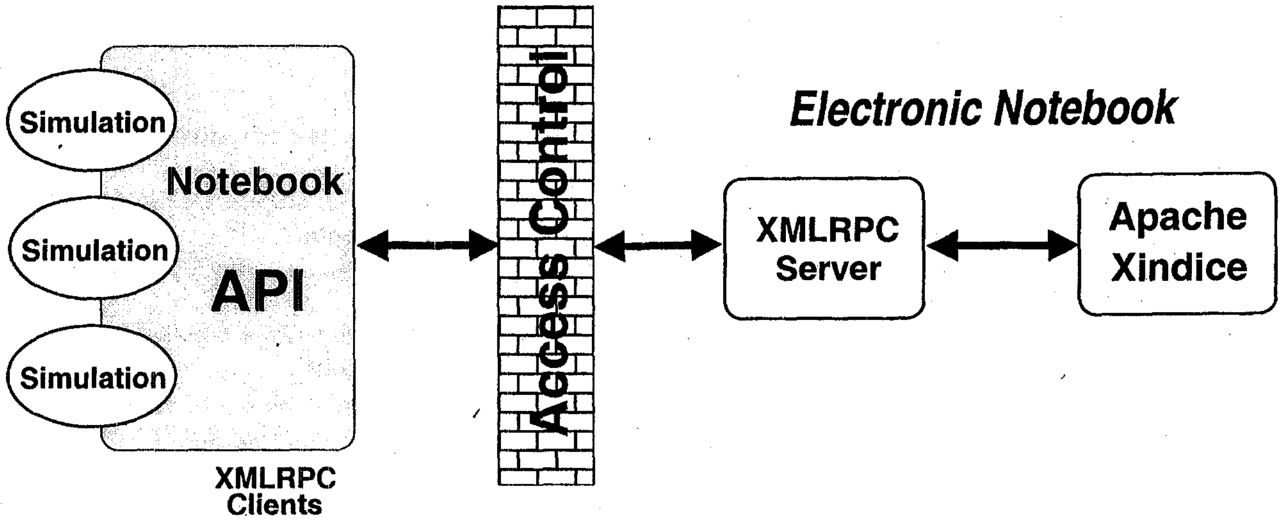

Figure 4. Simulation requests with the electronic notebook.

The notebook API actually consists of different forms of XMLRPC clients. The client makes a request

7. R. L. Kelsey, K. R. Bisset, and R. B. Webster, "Automated parametric execution and documentation for large-scale simulations," in *Enabling Technology for Simulation Science V*, **4367**, pp. 202–208, Society of Photo-Optical Instrumentation Engineers, Society of Photo-Optical Instrumentation Engineers, (Bellingham, WA), 2001.
8. R. L. Kelsey, J. M. Riese, and G. A. Young, "XML-based knowledge management for software development," in *Proceedings of the International Conference on Knowledge Engineering 2002*, 2002 International MultiConference, June 2002.
9. The Apache XML Project, *Apache Xindice*, 2002. http://xml.apache.org/xindice.
10. The Apache Jakarta Project, *Apache Tomcat*, 2002. http://jakarta.apache.org/tomcat.
11. Sun Microsystems, Inc., *Sun Java Technology*, 2003. http://java.sun.com.
12. Sun Microsystems, Inc., *Java Servlet Technology*, 2003. http://java.sun.com/servlet.
13. Sun Microsystems, Inc., *Java Server Pages Technology*, 2003. http://java.sun.com/jsp.
14. Userland Software, Inc., *XMLRPC Specification*, 2002. http://www.xmlrpc.com/spec.